

Finding a Hamiltonian cycle by minimizing a determinant

M. Haythorpe and W. Murray

Abstract It has been shown that the global minimizer of a smooth determinant of a matrix function reveals the largest cycle of a graph. When it exists this is a Hamiltonian cycle. Finding global minimizers even of a smooth function is a challenge. The difficulty is often exacerbated by the existence of many global minimizers. One may think this would help but in the case of Hamiltonian cycles the ratio of the number of global minimizers to the number of local minimizers is typically astronomically small. We describe efficient algorithms that seek to find global minimizers. There are various equivalent forms of the problem and here we describe the experience of two. The matrix function contains a matrix $P(x)$, where x are the variables of the problem. $P(x)$ may be constrained to be stochastic or doubly stochastic. More constraints help reduce the search space but complicate the definition of a basis for the null space. Even so we derive a definition of the null space basis for the doubly stochastic case that is as sparse as the constraint matrix and contains elements that are either 1, -1 or 0. Such constraints arise in other problems such as forms of the quadratic assignment problem.

Keywords: Hamiltonian cycle, barrier functions, interior-point methods, negative curvature.

1 Introduction

Given an undirected graph Γ containing N nodes, determining whether any simple cycles of length N exist in the graph solves the Hamiltonian cycle problem. Simple cycles of length N are known as Hamiltonian cycles. This paper is concerned with finding a Hamiltonian cycle (HC) of a graph by finding a global minimizer of a smooth function. We associate a variable x_{ij} with each (directed) arc $(i, j) \in \Gamma$. Define a matrix $P(x)$, whose (i, j) th element is x_{ij} if $(i, j) \in \Gamma$,

M. Haythorpe
Flinders University, Adelaide, Australia
E-mail: michael.haythorpe@flinders.edu.au

W. Murray
Stanford University, Palo Alto, US
E-mail: walter@stanford.edu

or 0 otherwise. It was shown in [1] that a longest cycle of a graph is a global minimizer of the problem:

$$\begin{aligned} & \underset{x}{\text{minimize}} && f(x) \equiv -\det(I - P(x) + \frac{1}{N}ee^T) \\ & \text{subject to} && P(x) \in \mathcal{S}, \quad x \geq 0, \end{aligned} \tag{1}$$

where \mathcal{S} is the set of stochastic matrices. We shall refer to the linear constraints that arise from this restriction on $P(x)$ as the \mathcal{S} constraints. It follows we may also restrict $P(x) \in \mathcal{DS}$, where \mathcal{DS} is the set of doubly stochastic matrices, since if a HC exists and the solution to (1) is defined to be x^* then $P(x^*)$ is a permutation matrix. It is these two forms of the problem that we investigate. The elements of $P(x^*)$ that are 1 denote the arcs in the HC. Although we introduced x as the nonzero elements of $P(x)$, in practice x is a vector. We number the indices by row. So if there are 3 arcs from the first node there will be x_1 , x_2 and x_3 in the first row of $P(x)$. The first nonzero element of $P(x)$ in the next row is x_4 and so on. $P(x)$ is symmetric in the pattern of the nonzero elements but is not a symmetric matrix. The elements in the upper triangular half correspond to arcs in one direction and their lower triangular half reflection is the arc being taken in the opposite direction. There would be no reason not to label arcs so that the (1,2) element was always nonzero. The corresponding (2,1) element must be zero if P_{12} at the solution is nonzero. However, it is possible for both to be zero.

When $P(x) \in \text{Int}(\mathcal{DS})$ it is possible to replace the objective function in (1) by the negative determinant of the leading principal minor of $I - P$. The result follows from among other things that the restrictions we place on x and $P(x)$ ensure the LU factorization of $I - P(x)$ exists without the need to permute the rows or columns. Also $I - P(x)$ has rank $N - 1$ and the leading principal minor is full rank. The proof was given in [3]. Unfortunately this does not hold when $P(x) \in \text{Int}(\mathcal{S})$. Using the leading principal minor has the advantage that the rank-one modification $\frac{1}{N}ee^T$ is not required, which makes calculating the gradient and the Hessian a little simpler. A method for efficiently computing the gradient and Hessian of the negative determinant of the leading principal minor was provided in [3], and was proved in [7] to be more numerically stable than the objective function in (1). Another benefit is that the maximum value is independent of the size of the graph, eliminating the need to scale any parameters by the size of the graph.

In the \mathcal{DS} case the problem of interest is of the form

$$\underset{x}{\text{minimize}} f(x)$$

subject to

$$\sum_{j \in \mathcal{A}(i)} x_{ij} = 1, \quad i = 1, \dots, N, \tag{2}$$

$$\sum_{i \in \mathcal{A}(j)} x_{ij} = 1, \quad j = 1, \dots, N, \tag{3}$$

$$x_{ij} \geq 0, \quad \forall (i, j) \in E, \tag{4}$$

where $f(x) = -\det(M)$, M is the leading principle minor of $I - P$, $\mathcal{A}(i)$ is the set of nodes reachable in one step from node i . Constraints (2)–(4) are called the *doubly-stochastic* constraints. For neatness, we refer to constraints (2)–(4) as the \mathcal{DS} constraints. It is assumed that any graph considered is simple and undirected. Although this is a classical linearly constrained problem it is different in character from those whose variables are not related to a binary-variable problem. A consequence of the multiple global minimizers (and we believe this is true in many other

problems in discrete variables such as assignment problems that are relaxed) is the presence of saddlepoints that are almost minimizers. Indeed potentially the number of saddlepoints can be much larger than the number of global minimizers. It can be shown that there exists a path between any two isolated global minimizers that contains a feasible saddlepoint. Moreover, there exists a saddlepoint for which that has only one negative eigenvalue. There is a potential for the number of such points to grow exponentially with the number of global minimizers.

It was shown in [3] how to compute $f(x)$ and its first and second derivatives very efficiently. This is critical since we show that directions of negative curvature are essential to solving this problem and they play a much more critical role than is typically the case. A key issue is symmetry. Obviously for *every* HC there is a HC obtained by reversing the direction. This symmetry reveals itself in the problem variables. If there is a HC with $x_1^* = 1$ then there exists a reverse cycle that is also a HC in which $x_1^* = 0$ and its twin is 1. We need to set the initial value of these two variables to be identical in order not to introduce bias (they may both be 0 in another HC). Quite frequently (and this almost always happens with some pairs) when using only descent many of these twin variables remain equal. In such circumstances it is only the use of a direction of negative curvature that breaks the tie. While such behavior is possible for general problems it is quite rare. Consequently, in this class of problem directions of negative curvature play a more important role and often more important than that of using a direction of descent. Again unlike the general case where we usually observe no directions of negative curvature in the neighborhood of the solution here they are always present, which is one reason why the solution is at a vertex. What is happening is that from our current iterate there are two equally attractive minimizers so it steers a course going to neither unless directions of negative curvature are used. The symmetry reveals itself also in the problem function and derivatives. If the iterates to solve the problem are denoted by $\{x_k\}$ then at x_k it is usually the case that g_k the gradient of $f(x)$ at x_k is orthogonal to the eigenvectors corresponding to negative eigenvalues of the Hessian of $f(x)$ at x_k . Consequently when are solving for the Newton step using the conjugate gradient algorithm it will not be detect when Hessian is indefinite.

2 Choice of constraints

Typically in an optimization problem it is better to have more constraints if such constraints can be added even if these are inequalities and are known not to be active at the solution. However, it is not always the case. We have a choice of either $P \in \mathcal{DS}$ or $P \in \mathcal{S}$ (eliminate equations 3 or 4). Note here when solving with $P \in \mathcal{DS}$ we are adding more equality constraints without adding extra variables and hence we are reducing the degrees of freedom in the problem. However, for this particular problem there are some theoretical differences that alter the usual picture of potentially reducing the search space but adding to the complexity of computing the iterates. It was shown in [1] that when $P \in \mathcal{DS}$ that the LU factors of $I - P$ exist regardless of the pivoting order. This has many beneficial consequences not the least of which is the objective of the problem may be recast to be $-\det(M)$, where M is the leading principle minor of $I - P$. Although $I - P$ is singular its leading principle minor is nonsingular as a consequence of the existence of the LU factorization. Note that $I - P$ is typically very sparse (if it is not finding a HC is usually trivial). It also has other benefits, the main one being that it enables the problem to be recast in a wide variety of ways. We shall show in section 3 that this property is also true even when $P \in \mathcal{S}$ so this is not a reason for preferring $P \in \mathcal{DS}$. In [1] it was shown as part of the proof that when $P \in \mathcal{S}$, if a variable is not 0 or 1 altering it to one of them reduces the objective. One consequence is that all local minimizers are binary. It has not been shown that this result is true for $P \in \mathcal{DS}$. Indeed it seems likely it is not true. The issue that makes it more

complicated is that altering a variable in the $P \in \mathcal{DS}$ usually requires altering many or all of the other variables in order to retain feasibility. There are many ways that could be done. It will be seen that one of the steps we propose in our algorithm is deletion or deflation, which occur when one or more of the variables is set to 0 or 1 respectively. For the $P \in \mathcal{S}$ case it is simple to adjust the corresponding variables in a row of P to satisfy the constraint simply by scaling the relevant row. For the $P \in \mathcal{DS}$ case it more complex and either an LP or QP needs to be solved. It is made more complicated by the need to determine a strictly interior point and sometimes one does not exist. Usually one benefit of more constraints is the reduced Hessian is smaller and the linear system needed to be solved at each iteration is also smaller.

Finding both a sufficient descent direction and a direction of sufficient negative curvature requires finding a null space matrix. If A is the constraint coefficients we require a matrix Z such that $AZ = 0$ and the matrix $(A^T \ Z)$ is full rank. The matrix Z is almost always dense. Consequently, the smaller the dimension of Z the better. However, it was shown in [3] that there exists a Z for the \mathcal{DS} case that is sparse and structured. Paradoxically the larger Z for the \mathcal{S} case is simpler and sparser. This alters the balance when computing the search directions needed to solve our problem.

3 Preliminaries

We show that the LU factorization of $I - P$ and of $I - P^T$ exists when P is a stochastic matrix. As already noted this was shown to be true for a doubly stochastic matrix. To determine the determinant of the objective we need to compute the LU factorization of a matrix and this result implies no pivoting is required.

A stochastic matrix may have either rows or columns that sum to unity. In forming the LU factorization it is common to assume row interchanges rather than column interchanges. This is just convention and there is no advantage to doing it one way or the other. However, for sparse matrices the manner the sparse elements are stored does matter when performing the LU factorization. Since when forming such matrices it is assumed that row interchanges will be done that impacts how best to store the sparse matrix in compact form. In the proof we assume row interchanges may be made and this causes us to prefer to assume that P has unit columns. The converse result for unit rows follows immediately from this result.

Definition 1 A matrix A is said to have property S_c if

$$\begin{aligned} A_{i,i} &\geq 0 \text{ for } \forall i \\ A_{i,j} &\leq 0 \text{ for } \forall i \neq j \\ A^T e &= 0. \end{aligned}$$

Theorem 1 If A has property S_c an LU factorization of A exists.

Proof

If $A_{1,1} = 0$ then the first row of L is e_1^T and the first row of U is the same as the first row of A . Consequently, there is no loss of generality if we assume that $A_{1,1} \neq 0$. Note that $A_{1,1}$ is

the element of largest magnitude in the first column of A . After one step of standard Gaussian elimination (GE) we get

$$A = \begin{pmatrix} 1 & 0 \\ l & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \hat{A} \end{pmatrix} \begin{pmatrix} U_{1,1} & u^T \\ 0 & I \end{pmatrix}.$$

We have $A^T e = 0$, which implies, since $A_{1,1} = U_{1,1} \neq 0$ that

$$e^T \begin{pmatrix} 1 & 0 \\ l & I \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \hat{A} \end{pmatrix} = 0$$

It follows that $\hat{A}^T e = 0$ and $e^T l = -1$. By definition we have

$$\hat{A}_{i,j} = A_{i+1,j+1} - l_i u_j \quad \forall \quad i \neq j.$$

Since $l_i \leq 0$ and $u_j \leq 0$ it follows that $\hat{A}_{i,j} \leq 0 \quad \forall \quad i \neq j$. From this result and $\hat{A}^T e = 0$ it follows that $\hat{A}_{i,i} \geq 0$ and that \hat{A} has property S_c . We can now proceed with next step of GE. Note that if $\hat{A}_{1,1} = 0$ we must have the first row and column of \hat{A} be zero and we can skip the steps of GE until we have a nonzero diagonal element of \hat{A} . ■

Corollary 1 *Regardless of the rank of A we have*

$$U_{n,n} = 0.$$

This follows from \hat{A} having property S_c and the only 1×1 matrix (the size of \hat{A} for the last step of GE) with that property being 0.

Corollary 2 *If A has rank $N - 1$ then $L^T e = e_N$.*

Corollary 3 *If A has rank $N - 1$ the leading principle minor is nonsingular.*

Corollary 4 *When performing GE the elements being eliminated are not larger in magnitude than the pivot. This implies that $0 \geq L_{i,j} \geq -1$.*

This property implies that L is about as well conditioned as it can be and that if software to perform GE is used even if it performs row interchanges when needed they will never be required and the LU factorization of A will be obtained and not that of PA , where P is a permutation matrix.

Lemma 1 *The matrix $B \equiv A + \beta e e_N^T$ is nonsingular when A has rank $N - 1$ and $\beta \neq 0$.*

Proof

We have

$$B = LU + \beta e e_N^T = L(U + \beta y e_N^T) = L U_B,$$

where $Ly = e$. Note that U_B is upper triangular. Moreover, the (N, N) th element of U_B is βy_N . Since $e^T L = e_N^T$ we get $y_N = N$, which implies B is nonsingular. ■

Definition 2 A matrix A is said to have property S_r if

$$\begin{aligned} A_{i,i} &\geq 0 \text{ for } \forall i \\ A_{i,j} &\leq 0 \text{ for } \forall i \neq j \\ Ae &= 0. \end{aligned}$$

Corollary 5 If A has property S_r an LU factorization of A exists.

This follows from the fact that the LU factors of A^T exist. LU factors of A can be obtained from the transpose of these factors. Note that although this is an LU factorization it differs from that typically found since it is now U that has unit diagonal elements.

4 Finding the global minimizer of the linearly constrained problem

The basic approach used is similar to that due to Murray and Ng [8], who first relax the problem and then solve a sequence of problems in which a strictly convex function is added to the objective together with a nonconvex function that attempts to force the variables to be binary. Initially the strictly convex function dominates the objective and in the limit the nonconvex term dominates. Our approach is a simplification since the nonconvex term is not needed. Also since we are applying this general approach to a specific problem with significant structure the algorithm can be modified to improve not only efficiency, but also to improve the likelihood of obtaining a global minimizer and hence a HC. How the individual problems in the sequence are solved is the main focus. It will be seen a much heavier use of negative curvature is made and with less emphasis on the use of descent directions, which is the reverse of what optimization algorithms usually do. A peculiarity of the problem, which we think may be true of most problems with multiple global minimizers, is the gradient at the iterates is often spanned by the eigenvectors corresponding to the positive eigenvalues of the Hessian even though the Hessian is indefinite. This corresponds to the so-called “hard case” in trust region methods. Typically in such methods little or no attention is paid to it since it is considered very unlikely to arise and essentially impossible to keep arising.

In both the stochastic and doubly stochastic case we are interested in solving a problem of the form:

$$\begin{aligned} \min f(x) \\ \text{s.t.} \end{aligned} \tag{5}$$

$$\begin{aligned} Ax &= e, \quad i = 1, 2, \dots, m, \\ 0 &\leq x \leq e. \end{aligned} \tag{6}$$

Strictly speaking the upper bounds on x are not required since the equality constraints and the lower bounds ensure that the upper bound on x holds. However, for now we shall leave them in.

We are interested in the global minimizer and a typical descent algorithm will converge to the local minimizer associated with the initial point. Murray and Ng propose adding a strictly convex function $\mu\phi(x)$ to the objective, where μ is a positive scalar. A sequence of problems is then solved for a sequence of strictly monotonically decreasing values of μ . For $\phi(x)$ with certain continuity properties the trajectory of minimizers $x^*(\mu)$ is a unique, continuous, and smooth trajectory.

When the initial value of μ is sufficiently large the new objective is also strictly convex and has a unique and therefore global minimizer. Consequently, the minimizer found by this algorithm is the one whose trajectory is linked to the initial unique global minimizer.

A feature of the problem is it has what we term “twin variables”. In the definition of the variables as elements of the matrix P , if P_{ij} is not always zero then neither is its twin P_{ji} . In terms of the graph this is the same edge except in the opposite direction. Since the reverse of a HC is itself a HC twin variables have an equal probability of being in a HC. It is essential that the minimizer of $\phi(x)$ is a neutral point with regard to the minimizers of the original problem. For example, if the feasible region is a hypercube the unique neutral point is the center. Since we know the binary minimizers are extreme points of the feasible region the “center” of the feasible region is such a point. One way of achieving such a point is to choose:

$$\phi(x) = -\sum_{i=1}^n \ln x_i.$$

An alternative is

$$\phi(x) = -\sum_{i=1}^n \ln x_i + \ln(1 - x_i).$$

These are well known barrier functions used in interior point methods. By using either of these functions we have transformed the original problem into minimizing a sequence of barrier functions. Note the reason here for using such functions is not eliminating inequality constraints, that is simply a side benefit. Solving the original problem using say an active set method is efficient especially since we do not expect the size of the problems to be extremely large (100,000 variables or more). Our motivation is different and consequently it impacts how the initial μ is chosen and how it is subsequently adjusted. Since we seek a neutral point $\mu_0 = \infty$ ($f(x)$ dropped from the objective). A test of whether the choice of $\phi(x)$ leads to a neutral initial point is whether the twin variable have the same initial value and this is observed in the numerical testing of the barrier function.

Since the barrier function removes the need for the inequality constraints the algorithms requires the solution of a sequence of linearly *equality* constrained optimization problems.

4.1 Solution of the linearly equality constrained subproblem

The choice of method is dictated by the need to converge to points that at least satisfy second-order optimality conditions. This requires the algorithm to determine whether the reduced Hessian is positive semidefinite. To obtain the reduced Hessian matrix we need the null space matrix Z , which is such that $AZ = 0$ and $(A^T Z)$ is full rank. The reduced Hessian is then given by $Z^T H Z$, where H is the Hessian of $f(x)$.

We use a line search method based on determining a descent direction and when available a direction of negative curvature. A sequence $\{x_k\}$ of improving estimates is generated from an initial feasible estimate x_0 from

$$x_{k+1} = x_k + \alpha_k(p_k + d_k),$$

where α_k is a steplength that ensures a sufficient decrease, p_k is a sufficient descent direction, and d_k is a direction of sufficient negative curvature. It was shown by Forsgren and Murray [4] that this sequence converges to a point that satisfies the second-order optimality conditions.

Typically such methods combine a direction of descent with a direction of negative curvature when the latter exists. Our observation is when a direction of negative curvature does exist and is used purely as the search direction then at almost every subsequent iteration a direction of negative curvature exists and is usually getting stronger. Consequently, when we get a direction of negative curvature we do not bother computing the direction of descent.

Given the importance of the direction of negative curvature we depart from normal practice and apply the modified Cholesky algorithm [5,6] to the following matrix

$$Z^T H Z - \delta I,$$

where δ is an estimate of the smallest eigenvalue of $Z^T H Z$ when it is thought $Z^T H Z$ is indefinite, otherwise δ is negative and very small in magnitude. The rational is that when $Z^T H Z$ is indefinite this leads to a very good direction of sufficient negative curvature. When $Z^T H Z$ is positive definite the small shift ensures that the matrix has a condition number that is sufficiently small to ensure sufficient accuracy in the direction of descent. If no modification is made in the modified Cholesky factorization then $Z^T H Z - \delta I$ is positive definite and we compute a direction of sufficient descent by solving:

$$R^T R p_z = -Z^T g,$$

where R is the upper triangular factor, and g is the gradient of $f(x)$. The sufficient descent direction is then given by p , where $p = Z p_z$. If a modification is made then $Z^T H Z - \delta I$ is indefinite and the following system is solved

$$R d_z = e_j,$$

where the index j is obtained during the modified Cholesky factorization. It can be shown that d , where $d = Z d_z$ is a direction of sufficient negative curvature. Moreover, we have

$$d^T H d \leq \delta d^T d.$$

We can improve this direction of negative curvature by minimizing $d^T H d / d^T d$. We reduce the value by doing a sweep of univariate minimization of this function. This cost is roughly the same as a matrix-vector multiplication and so can be repeated if need be. We use the improved value as the estimate of δ in the following iteration. Note that the sign of d is always chosen so that $d^T g \leq 0$.

If δ is not small in magnitude we will not know if negative curvature exists when no modification is made in the modified Cholesky algorithm. However we will know that the smallest eigenvalue is bigger than δ . We repeat the modified Cholesky factorization with $\delta \leftarrow \delta/2$. If after a small number of reductions we still get no modification then δ is set to the default small value.

We use a very crude linesearch along either p or d . We compute the maximum step to the boundary and take a step α times the value. If that is not a lower point we multiply the step by 0.5 until we succeed. Typically $\alpha = 0.9$ and almost always is successful.

5 The outer algorithm

A key difference with the use of a barrier function here compared to solving problems unrelated to relaxed discrete problems is that $f(x)$ behaves in an unusual way. After a strictly feasible point is found this is used to minimize the barrier function alone (equivalent to setting $\mu = \infty$). This is an easy function to minimize and to do so accurately. This is necessary to avoid bias unlike when minimizing a regular function where we are often able to provide an initial point reasonably close to the solution. Indeed we are attempting to find the initial point to be as far away as possible from the solutions. In some cases such as for cubic graphs the minimizer of the barrier function is known ($x_i = 1/3$). Typically we want to reduce μ at a slow rate. However, another feature of the HC problem is the point that minimizes the barrier function is either a saddle point of the determinant function or very close to it. Again this rarely if ever happens when using a barrier function for normal problems. The consequence is that moving the iterates from their current location requires changing μ sufficiently to make the current reduced Hessian indefinite. Quite how much is not difficult to estimate. The Hessian of the barrier function is a well conditioned diagonal at the minimizer. It is usually less than 2 and for cubic graphs is 1. In both the stochastic and doubly stochastic case the matrix Z has a low condition number. Consequently, given an estimate of the smallest eigenvalue of either H_D , the Hessian of determinant function, or of $Z^T H_D Z$ it is easy to find a good estimate of the change needed in μ . If it is not sufficient then we can simply divide by 10 until it is. In our testing this was never needed. Once we get negative curvature we usually never reduce μ again since either we succeed in finding a HC without needing to, or we fail.

An alternative to solving the standard problem is to use the primal dual approach. The standard approach means that the Newton direction is poor when μ is small. There are two reasons not to use the primal dual method. Firstly, we do not need to have μ very small since we know the solution is converging to a binary point and so we can round and test the solution. Ill-conditioning arises due to a variable becoming close to a bound. Should that happen such a variable can be removed from the problem. How to do this is described in the sections on deletion and deflation. Secondly, we need to use directions of negative curvature but the Hessian in the primal dual formulation is not assured to give a direction of negative curvature except in the neighborhood of a stationary point.

Definition of the null space Z

A common way of defining Z , such that $AZ = 0$ and $(A^T Z)$ is full rank, is to first partition the columns of $A = [B \ S]$, where B is nonsingular. Then we can define

$$Z := \begin{bmatrix} -B^{-1}S \\ I \end{bmatrix}.$$

If only stochastic constraints are required, the matrix A can be quite simply defined. If the graph has N vertices, where vertex i has degree d_i , then we can define

$$A := [A_1 \ A_2 \ \dots \ A_N],$$

where $A_i = e_i e^T$ is an $N \times d_i$ matrix. It is easy to see that the condition number of A is bounded above by N , by checking that $AA^T = \text{diag}[d_1 \ d_2 \ \dots \ d_N]$, and therefore the condition number of A is the ratio of the largest degree to the smallest degree.

In order to define the null space matrix for A , it is trivial to reorder the columns of A such that the first column of each A_i submatrix appears first. The reordered matrix is $\hat{A} := [I \ S]$, where S is defined as

$$S := [S_1 \ S_2 \ \dots \ S_N],$$

and $S_i = e_i e^T$ is an $N \times (d_i - 1)$ matrix. Then the null space matrix for \hat{A} is

$$\hat{Z} = \begin{bmatrix} -S \\ I \end{bmatrix},$$

and appropriately reordering the rows of \hat{Z} provides the null space matrix for A . One advantage of defining the null space matrix in this way is that the sparsity inherent in difficult instances is retained in Z , and non-zero entries are all either +1 or -1. The condition number of Z is equal to $\max_i d_i$. The only operations involving Z that are required are matrix-vector products. Then for a given vector v , the product Zv can be computed very efficiently.

If doubly-stochastic constraints are desired, the matrix A defines constraints on both the rows of P and the columns of P . We first define a matrix A_r , corresponding to the row constraints, to be identical to the A matrix for the stochastic case. Next we define a matrix A_c , corresponding to the column constraints. Suppose each variable x_k corresponds to an arc $a_k = (i, j)$. Then A_c is defined as

$$[A_c]_{jk} := \begin{cases} 1 & \text{if } \exists i \text{ s.t. } a_k = (i, j) \\ 0 & \text{otherwise} \end{cases}$$

Then, we can define A to be

$$A := \begin{bmatrix} A_r \\ A_c \end{bmatrix}.$$

The matrix A defined in this way is certain to be rank deficient. In order to construct a null space matrix, we want to delete enough rows to obtain a full rank matrix, and then reorder the matrix to obtain $\hat{A} = [B \ S]$, where B is a *triangular* matrix. This can be achieved by using the following algorithm.

```

Input:  $A, N$ 
Output:  $B, S, \mathcal{I}$ 

begin
  count  $\leftarrow 0$ 
  rows  $\leftarrow \text{rank}(A)$ 
   $\hat{A} \leftarrow A$  with rows removed to make  $\hat{A}$  full rank
  cols  $\leftarrow \text{columns}(\hat{A})$ 
   $r \leftarrow \text{rows}$ 
   $\mathcal{I} \leftarrow \{1, \dots, \text{cols}\}$ 
  while  $r > 0$ 
     $C \leftarrow$  Identify a set of columns  $\{c_1, \dots, c_k\}$  such that  $\sum_{i=1}^r a_{ic_j} = 1, \quad \forall j = 1, \dots, k$ 
    and  $a_{ic_j} a_{ic_k} = 0, \quad \forall i = 1, \dots, r, \quad j \neq k$ 
    for  $i$  from 1 to  $k$ 
      count  $\leftarrow \text{count} + 1$ 
       $\mathcal{I} \leftarrow [\mathcal{I}_1 \dots \mathcal{I}_{\text{count}-1} \mathcal{I}_i \mathcal{I}_{\text{count}} \dots]$  (Moving  $\mathcal{I}_i$  into position count)
       $\hat{A} \leftarrow \hat{A}(\mathcal{I})$  (Moving column  $c_i$  to column count)
       $\hat{A} \leftarrow$  reorder the rows to get a 1 in positive  $(r - i + 1, \text{count})$ 
    end
     $r \leftarrow \text{rows} - \text{count}$ 
  end
   $\mathcal{I} \leftarrow$  Reverse the order of the first rows entries in  $\mathcal{I}$ 
   $\hat{A} \leftarrow \hat{A}(\mathcal{I})$  (Reverse the order of the first rows columns in  $A$ )
   $B \leftarrow \hat{A}(1 : \text{rows}, 1 : \text{rows})$ 
   $S \leftarrow \hat{A}(1 : \text{rows}, \text{rows}+1 : \text{cols})$ 
end

```

Reordering A algorithm

Then the null space for \hat{A} is defined to be

$$\hat{Z} := \begin{bmatrix} -B^{-1}S \\ I \end{bmatrix}.$$

Unlike typical problems the Z constructed in this way is sparse (similar to that of A) and does not require the LU factorization of B since B is lower triangular. Moreover, B has elements that are either 0 or 1. Consequently operations with Z do not require any multiplication.

Deletion and deflation

If at any stage, one or more of the x_{ij} variables approach their extremal values (0 or 1), we fix these values and remove the variables from the problem. This process takes two forms: *deletion* and *deflation*, that is, setting x_{ij} to 0 or 1, respectively. Note that we use the term deflation because in practice the process of fixing $x_{ij} := 1$ results in two nodes being combined to become a single node, reducing the total number of nodes in the graph by 1.

Deletion is a simple process of fixing a variable to 0 by simply removing its associated arc from the graph. When a variable x_{ij} is close to 1, we perform a deflation step by combining nodes i and j by removing node i from the graph. Then, we redirect any arcs (k, i) that previously went into node i to become (k, j) , unless this creates a self-loop arc. After deletion or deflation we construct the new constraint matrix A and update Z appropriately. The thresholds for the deletion or deflation process to take place can be set as input parameters.

During deflation, we not only fix one variable (x_{ij}) to have the value 1, but also fix several other variables to have the value 0. Namely, we fix all variables corresponding to arcs (i, k) for $k \neq j$,

(k, j) for $k \neq i$, and (j, i) to have the value 0. Whenever we perform deflation the information about the deflated arcs are stored in order to construct a HC in the original graph once a HC is found in the reduced graph.

After performing deletion or deflation, a reduced vector \bar{x} is obtained, which is infeasible in the resultant smaller dimension problem. In the stochastic case obtaining a feasible point is trivial since only the variables in the specific rows where fixing has occurred need to be adjusted. The simplest way is to multiply the remaining variables in an impacted rows by $(1/(1 - \bar{x}_{ij}))$, where \bar{x}_{ij} is the variable that has been fixed. Note that this increases the remaining variables so will not trigger another deletion in the row. It is possible it triggers a deflation, but this is unlikely. In the doubly stochastic case many or all of the variables may be impacted even for a single variable being deleted. Define $s := e - A\bar{x}$ to be the error induced by such a process. Note that s is a nonnegative vector in the case of both deletion and deflation. Then, we find a new x such that $x \in \mathcal{DS}$, and $|x - \bar{x}| < \varepsilon$, where the size of ε depends on the deletion or deflation thresholds chosen. The interpretation of x is that it is a point that satisfies the \mathcal{DS} constraints, and is as close as possible to the point we obtained after deleting or deflating.

We determine x by first defining $v \geq 0$ and $u \geq 0$ so that $x = \bar{x} + v - u$. Define x_{min} as the smallest element of \bar{x} . Then, we solve

$$\min_{u, v, \gamma} \rho\gamma + e^T(u + v) \quad (7)$$

s.t.

$$Au - Av + \gamma s = s, \quad (8)$$

$$x_{min} - \bar{x} \leq u - v \leq 1 - \bar{x}, \quad (9)$$

$$0 \leq u, v, \gamma \leq 1, \quad (10)$$

where ρ is chosen large enough that γ is reduced to 0 whenever possible. Constraints (9) are designed to ensure that $x \in \text{Int}(\mathcal{DS})$. However, it may be impossible to satisfy the above constraints for a value of $\gamma = 0$ because some variables may need to be 0 or a value very close to 0. In this case, we reduce x_{min} and solve the LP again, continuing this process until we obtain a solution with $\gamma = 0$.

If $\gamma \neq 0$ unless we set $x_{ij} = 0$ for some i and j , then we delete these variables, as they cannot be nonzero in a Hamiltonian cycle (or in fact any \mathcal{DS} point) containing the currently fixed arcs.

Rounding

At each iteration we test if a HC can be obtained by a simple rounding procedure. In the \mathcal{S} case we set the largest element in each row to one, starting with the row with the largest overall element. If the largest element happens to already have a unit element in the column we set the second largest in that row to one and so on. If there is no element available we have not identified a HC. After each setting of a variable to one we rebalance the constraints. A similar procedure is used in the \mathcal{DS} case except in this case setting an element to unity induces more elements being set to zero. Moreover, we now fail to satisfy the \mathcal{DS} constraints and so rebalancing is not done.

Obviously, we could use more sophisticated rounding methods, which may allow us to identify a HC earlier. One potential improvement of this method would be to solve a heuristic at the

completion of each iteration, using the current point \mathbf{x} , that tries to find a nearby HC. Such a hybrid approach was considered in [2], with promising results. This has not been explored since we are interested in testing our algorithm to the limit.

Below we outline the structure of the algorithm, which we term DIPA (Determinant Interior Point Algorithm)

```

Input:  $\Gamma$ 
Output: HC

begin
   $\mathbf{x} \leftarrow$  Find initial interior point
   $\mathbf{x} \leftarrow$  Find barrier point
   $\mu \leftarrow \mu_{initial}$ 
  while HC has not been found
    if  $\mathbf{x}$  is a local min of barrier function
      Reduce  $\mu$ 
      if  $\mu$  too small
        Return no HC found, converged to non-HC local min
      end
    else
      if reduced Hessian positive definite
         $\mathbf{d} \leftarrow$  Find descent direction
      else
         $\mathbf{d} \leftarrow$  Find direction of negative curvature
      end
       $\alpha \leftarrow$  Choose step length
       $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{d}$ 

      begin deflation/deletion check
        if any variables are above deflation threshold
          Perform deflation
        else if any variables are below deletion threshold
          Perform deletion
        end
        if any deletion or deflation occurred
          if graph is no longer connected
            Return no HC found
          else
             $x \leftarrow$  nearby feasible point
            Repeat deflation/deletion step
          end
        end
      end
    end
  end
  if  $\mathbf{x}$  rounded to HC
    Return HC
  end
end

```

DIPA algorithm.

6 Numerical experiments

In order to investigate the character and to test the performance of the algorithm we generated a test set of 350 problems. Specifically, we randomly generated 50 problems for each of 20, 30,

...., and 80 nodes with node degree between 3 and 6. The computer used for performing all the experiments was a PC with Intel® Core™ i7-4600U CPU, 2.70 GHz, 16 GB of RAM, and running on the operating system Windows 8.1 Enterprise. DIPA was implemented in MATLAB R2014b, with all LPs solved by IBM ILOG CPLEX Optimization Studio v12.6 via its Concert interface to MATLAB.

The choice of initial μ and the rate of reduction of μ did not prove to be critical. For successful runs once μ had been reduced to a sufficient level for the reduced Hessian to be indefinite it almost always remained indefinite until it stopped. Indeed no further reduction in μ was required. In Figure 1 we show how the determinant function behaves as it goes from the initial point to all of the HCs of a 20 node graph. It can be seen all curves are remarkably similar and that each can be reached by going down a direction of negative curvature. Also the degree of curvature increases the closer the point gets to the HC.

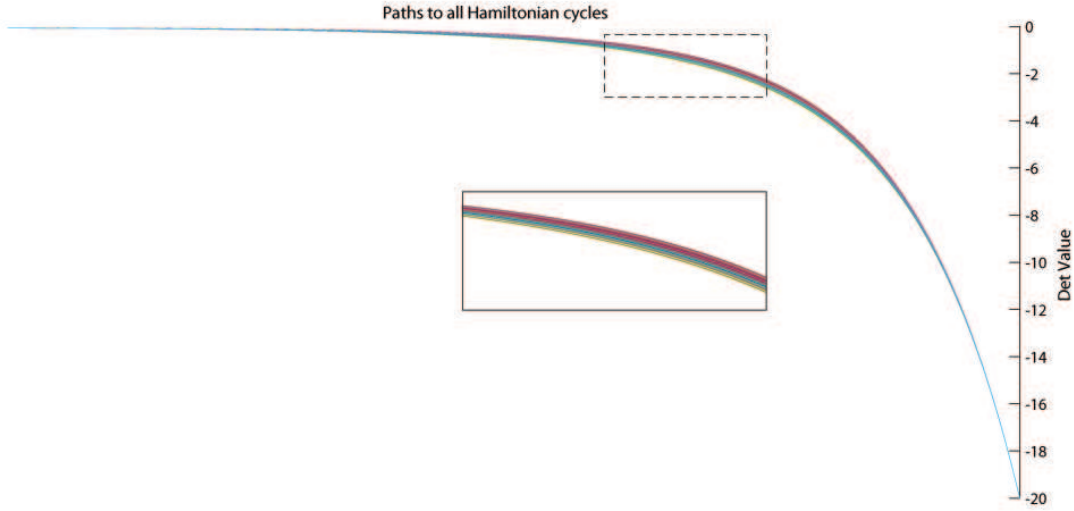


Fig. 1 Paths showing how the determinant function behaves as it goes from the initial point to all Hamiltonian cycles of a 20-node graph.

We attempted to solve the 350 problems in the test set with the algorithm applied to the stochastic and doubly stochastic form of the problem. In both cases an attempt was made to perform neither a deletion or deflation by setting extreme values for the relevant parameters that invoke these steps within the algorithm. The results given in Table 1 are unambiguous. It is clear that the doubly stochastic form of the problem is far superior. We did do further tests on the stochastic case by varying the adjustable parameters but the gap in performance was far too large to bridge.

N	20	30	40	50	60	70	80
Solved \mathcal{S}	26	19	13	9	10	7	9
Solved \mathcal{DS}	50	47	48	46	39	45	45

Table 1 Numbers of graphs solved when deletions and deflations are suppressed for the stochastic and doubly-stochastic forms of the problem.

A common technique in global optimization when a global minimizer is not found is to try random sets of initial points. That option is not open to us since a neutral initial point is required. We discovered that altering the adjustable parameters and options induced a large variance in which graphs were successfully solved. Since not all problems were solved we varied some of the options and adjustable parameters. We also introduced two other options. In addition to the use of an LP routine to obtain an interior point we also could use the CPLEX QP routine to find an interior point that minimized the Euclidean length from the current iterate. Since the step from the initial point to the solution of both the QP and LP is very small one would not expect this to have any impact on efficiency. Indeed, the infeasibility in the linear equalities is very tiny and this needs to be handled with care since CPLEX can simply treat it as being sufficiently small to be a solution. The other option was to remove one of the variables altogether. Doing so does not alter whether or not a graph has a HC since the reverse cycle exists. One would not expect removing one variable to have a measurable difference in performance. We were interested whether this would impact the set of graphs solved.

We ran the algorithm for several different settings of parameters and options. The algorithm typically ran just as efficiently for any alternative setting of the parameters and options chosen. Graphs that are not successfully solved can be attempted again with a different set of parameters. With the option to vary the use of the LP or QP or different deflation or deletion choices the attempt to obtain a solution can be made by restarting at the point this option is first used.

The adjustable input parameters and options are:

- Initial μ
- Amount by which μ is reduced when a local minimum is reached
- α - proportion of largest feasible step along the search direction
- Choose to use a LP or QP to find a feasible point after deletion or deflation.
- Deflation threshold
- Deletion threshold
- Use barrier function on $x \leq e$
- Remove one variable from the problem

The choice of α did have an impact on which graphs were solved, but setting α significantly less than one meant that the algorithm usually took longer to converge. In all results reported we set the initial $\mu = 0.01$, the reduction multiplier on μ to be 0.1, and $\alpha = 0.9$. It appears that deflation is more useful than deletion, so we set the deletion threshold to be very tiny, at 0.00001. In Table 2 we report the number of graphs from each test set that are solved for four settings. Specifically, we choose between obtaining an interior point via the LP or the QP, and we set the deflation threshold to be either 0.9 or 0.95. In Table 3 we also report the number of graphs that are solved after combinations of two of these approaches are employed, and then the final column after all four are employed.

In Table 4 we report similar results when the results for the case when no deflation is performed. Under such circumstances the option of whether to use an LP or QP is irrelevant. Instead, we vary whether or not to have the barrier term on the upper bound and whether or not to remove one variable. Table 5 reports the results of the combinations.

Not using deflation typically increased the number of iterations about 50% over using deflation. In Figure 2 we show how our algorithm typically converges when a HC is found. Since we are

N	Solved (LP, def=0.9)	Solved (LP, def=0.95)	Solved (QP, def=0.9)	Solved (QP, def=0.95)
20	49	50	50	49
30	50	49	47	46
40	45	46	42	43
50	42	41	46	45
60	45	39	39	40
70	40	36	40	35
80	40	39	32	36

Table 2 Numbers of graphs solved from each set of 50 for different choices of parameters.

N	LP, def 0.9 LP, def=0.95	QP, def=0.9 QP, def=0.95	LP, def=0.9 QP, def=0.9	LP, def=0.95 QP, def=0.95	Combined (all four)
20	50	50	50	50	50
30	50	49	50	50	50
40	49	49	49	50	50
50	50	50	50	49	50
60	48	49	50	49	50
70	46	46	46	44	49
80	48	43	47	47	50

Table 3 Spread of graphs solved over multiple runs with different choices of parameters.

N	No Upper Log No Removed Var	No Upper Log Removed Var	Upper Log No Removed Var	Upper Log Removed Var
20	50	49	50	50
30	49	49	49	47
40	48	45	48	48
50	49	47	47	46
60	43	45	46	39
70	47	44	44	45
80	45	44	41	45

Table 4 Numbers of graphs solved from each set of 50 for different choices of parameters, with deflation and deletion prevented.

N	Upper Log Combo	Removed Var Combo	No Upper Log Combo	No Removed Var Combo	Combined (all four)
20	50	50	50	50	50
30	50	50	50	49	50
40	50	49	49	48	50
50	50	49	50	49	50
60	49	48	50	48	50
70	49	50	50	48	50
80	48	50	49	47	50

Table 5 Spread of graphs solved over multiple runs with different choices of parameters, with deflation and deletion prevented.

converging to a vertex the nature of converges differs considerably from a typical minimization algorithm where the reduction made in the object slows as the solution is approached.

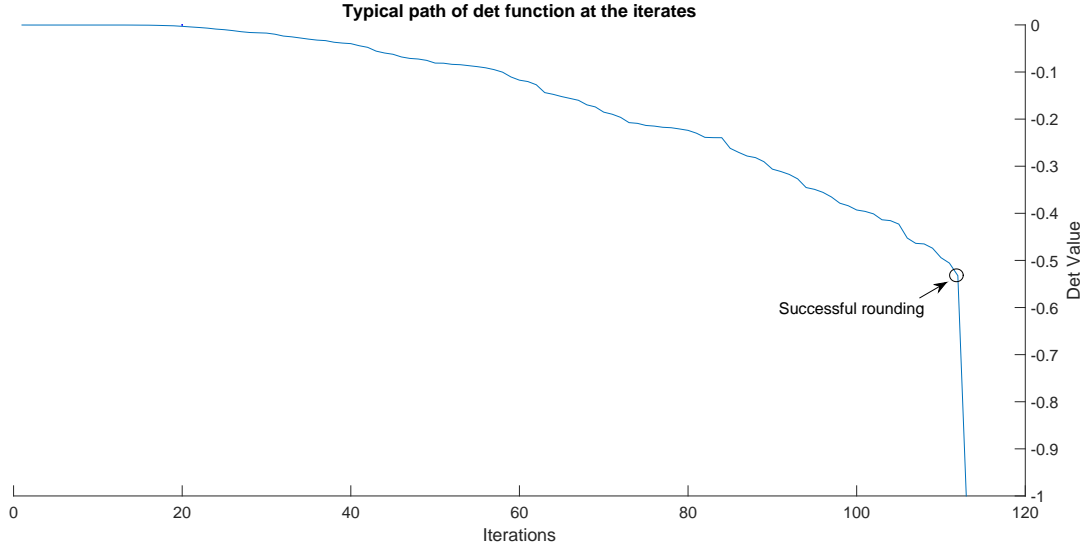


Fig. 2 A typical path of the determinant function at the iterates of the algorithm. After 112 iterations the rounding procedure finds a HC.

7 Concluding remarks

Clearly the results indicate that the problem with \mathcal{DS} constraints yields far better results than the problem with \mathcal{S} constraints. It also supports the view that different forms of a problem with identical complexity properties can have quite differing performance. It was not the intent of this paper to show or suggest that this approach is competitive with alternative algorithms to find a Hamiltonian cycle. However, it is quite distinct from other methods and there is much that can be done to improve its performance. More constraints can be added. For example, $P^T P = I$ and we know twin variables, say x_i and x_j must satisfy $1 - x_i - x_j \geq 0$ and $x_i x_j = 0$. The product from the latter constraints can be added directly to the objective and can be used initially to give a strictly convex problem. However, we plan to try and find a form of the problem that eliminates the occurrence of reverse cycles and so eliminates twin variables from the problem. The new variables would be the elements of Q , where $Q = 0.5(P + P^T)$. Knowing Q it is trivial to find the elements of P . The current problem has a dense Hessian matrix. Although we have shown how all the elements can be computed efficiently it still leaves a dense matrix, which has computational implications when computing the search direction for large problems. We are investigating transformations that should lead to the Hessian being sparse. Also if the conjugate gradient algorithm is used to compute the search direction and direction of negative curvature it may be possible to compute Hv efficiently even when H is dense.

Although we have addressed the Hamiltonian cycle problem an equally important interest is developing algorithms to determine global minimizers. In particular problems that have arisen from relaxing discrete problems. Many of the issues that arise in such problems are identical to those arising in the HC problem. For example, lots of global minimizers and hence lots of stationary points that have reduced Hessians that are almost positive definite (one negative eigenvalue). Moreover, symmetry is also present. Problems such as the frequency assignment problem have an equally good solution simply from any permutation of a known solution. Also solutions are typically at a highly degenerate vertex. We are encouraged by the success of the algorithm we have developed, which has demonstrated the ability to find global minimizers of highly nonlinear

and nonconvex problems with several hundred binary variables. A key requirement when solving problems with relaxed discrete variables is to have an unbiased initial point. As already noted a common technique used in global optimization is to use multiple starting points. The approach we advocate requires a neutral starting point. We have demonstrated that an equally good alternative is to vary some of the parameters and options that algorithms to solve such problems typically have. We have shown that very small changes both to the strategy and flexible parameters leads to distinct solution enabling us to reduce significantly the number of problems on which we fail to find a global minimizer. Moreover, these variations do not lead to less efficient methods.

References

1. V. Ejov, J.A. Filar, W. Murray and G.T. Nguyen. Determinants and longest cycles of graphs. *SIAM Journal on Discrete Mathematics*, 22(3):1215–1225, 2009.
2. A. Eshragh, J.A. Filar and M. Haythorpe. A hybrid Simulation-Optimization Algorithm for the Hamiltonian cycle problem. *Annals of Operations Research*, 189(1):103–125, 2011.
3. J.A. Filar, M. Haythorpe and W. Murray. On the Determinant and its Derivatives of the Rank-one Corrected Generator of a Markov Chain on a Graph. *Journal of Global Optimization*, 56(4):1425–1440, 2013.
4. A. Forsgren and W. Murray. Newton methods for large-scale linear equality-constrained minimization. *SIAM Journal on Matrix Analysis and Applications*, 14(2):560–587, 1993.
5. P.E. Gill and W. Murray. Newton-type methods for unconstrained and linearly constraints optimization. *Mathematical Programming*, 7(1):311–350, 1974.
6. P.E. Gill, W. Murray and M.H. Wright. *Practical Optimization*. London: Academic Press, 1981.
7. M. Haythorpe. Markov Chain Based Algorithms for the Hamiltonian Cycle Problem. PhD Thesis, University of South Australia, 2010.
8. W. Murray and K-M. Ng. An algorithm for nonlinear optimization problems with binary variables. *Computational Optimization and Applications*, 47(2):257–288, 2010.